Radio Astronomy VSR Interface (RAVI) Architecture and Protocol Definition

Requirements

The Radio Astronomy Group at JPL needs the ability to do some post processing of data recorded on the VLBI Science Receiver (VSR) at the station so that investigators and operators can view Radio Astronomy specific plots and analyze data, in real-time or near real-time, in a manner that the VSR does not or cannot provide. This will require another computer, which can be called the Radio Astronomy VSR Interface (RAVI), at the station which will download data files from the VSR, do the necessary post processing, and pass plots and other data to a client process (*exp_control*), the Radio Astronomy Controller (RAC), or other hosts.

The RAVI needs to be a relatively fast machine for processing large amounts of data and doing floating-point intensive operations such as FFTs. The RAVI needs a command interface for *exp_control* that specifies what plots to send, how many points and averages in the FFT etc… The RAVI will also need to provide an interface that allows *exp_control* to control the VSR for running the experiment that generates VSR data files to process.

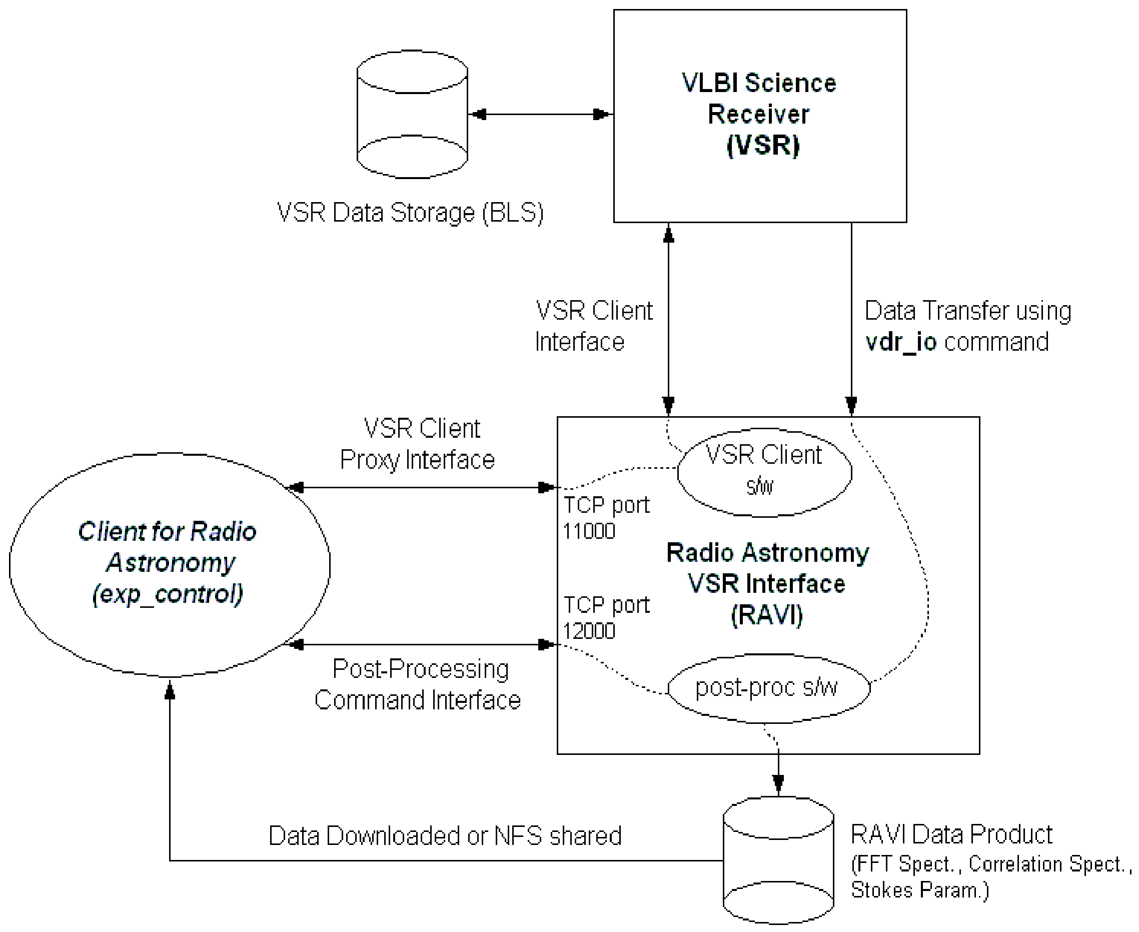
Basic Architecture

The diagram below shows the RAVI in relation to the VSR and *exp_control*. The RAVI is a multiprocessor Linux computer with 100MBits/sec access to the LAN and large amounts of disc space for storing data files from the VSR. To run an experiment, *exp_control* will open a connection, spawning a client on the RAVI. This client will be used a proxy for controlling the VSR by sending commands for configuration and data recording and receiving command responses. At the same time, *exp_control* will open another connection to the RAVI and send commands that control data file transfer from the VSR to the RAVI as well as specifying which parameters use in post-processing.

VSR Control

To control and run VSR experiments, *exp_control* first initiates a bi-directional socket connection on the RAVI using port 11000. This spawns a VSR text client (tclient). *exp_control* then passes a string of text providing station and VSR information to the tclient of the form:

“<dss_id> <vsr_id>”


VLBI Science Receiver (VSR)
VSR Data Storage (BLS)
VSR Client Interface
Data Transfer using vdr_io command
VSR Client Proxy Interface
Client for Radio Astronomy (exp_control)
VSR Client s/w
TCP port 11000
Radio Astronomy VSR Interface (RAVI)
TCP port 12000
Post-Processing Command Interface
post-proc s/w
Data Downloaded or NFS shared
RAVI Data Product (FFT Spect., Correlation Spect., Stokes Param.)

where: dss_id is 10, 11, 13, 21, 40 or 60
(w)vsr_id is vsr1a, vsr1b, vsr2a, vsr2b

which opens a connection to a specific VSR. A status message of the form:

“Client Connected”

is sent back to *exp_control* indicating that the VSR is ready to accept commands. Next *exp_control* sends a set of commands for running the VSR experiment using the VSR command language defined in section 2 of the Software Operators Manual or SOM (DSMS 837-037, Rev B). These commands include choosing the IF source, setting the ADC attenuation, setting the frequency, configuring the sub-channels and specifying the data file configurations.

The following example shows a simple set of commands that might be used to run a radio astronomy experiment:

```
IFS 14_X_RCP
ATT AUTO
FROV 8403456789
CHAN 1 100:8 0 1:2 0
CHAN 2 0 50:16 0 8:2
RECFN 1N1 vsr1a_100KHz_8bit
RECFN 1W1 vsr1a_1MHz_2bit
RECFN 2N1 vsr1a_50KHz_16bit
RECFN 2W2 vsr1a_8MHz_2bit
DDCLO ALL  303
RUN
FGAIN ALL AUTO
REC ALL ENABLE
+300 REC ALL DISABLE
HALT
QUIT
```

These commands choose the IF source DSS 14 at X-band RCP, set the front end attenuation, configure the VSR to point at a fixed RF frequency, configure 2 channels with sub-channels with one narrowband and one wideband each, set the local oscillators to automatically position the center of the channels to the specified fixed RF frequency and put the VSR into the RUN state. Data filenames are set for each sub-channel. Recordings are made of all configured sub-channels for 5 minutes after which the VSR is shut down. Responses to the commands, specified in section 4 of the SOM, are sent back to *exp_control* and can be parsed for status.

Post Processing Control

Opening a bi-directional socket on port 12000 of the RAVI establishes a command channel for controlling the post processing. *exp_control* can then send commands specifying how to process data files recorded using the VSR client proxy. The post processing data products are stored in files with filenames that can identify the content. These files can be NFS shared between the RAVI and *exp_control* and are in ASCII format to avoid any endian (byte order) issues between different platforms.

Post Processing Commands

The post processing software on RAVI consists of a command processor and multiple commands that can be run from the command processor interface over the socket. Some of the commands can be run as standalone applications from the unix command line of the RAVI. These include CORR, FFT and STOKES, all of which will plot with results with gnuplot as well as create ascii text files with data when run from the unix command line of the RAVI The command processor uses a '$' prompt to indicate a state ready to process commands. The commands currently implemented are:

HELP – Provide a list of commands
Usage:
HELP COMMAND = <string> - provide additional help for <string>

CORR – correlate two data sequences
Usage: CORR [-ahlt] vsr_host data_file1 data_file2 num_bin int_time
arguments:

-a	Align the sequences for maximum correlation (peak lag is centered)
-h	Use a Hanning window on the fft input data points
-l	Process VSR data files stored in local unix format
-t {time}	ASCII string in YY/DDD/HH:MM:SS format that specifies the time to start processing the data files (default: last record)
vsr_host	Name of VSR to get file(s) from
data_file1	Name of first BLS data file on the VSR to process
data_file2	Name of second BLS data file on the VSR to process
num_bin	Number of bins in the cross correlation spectrum
int_time	Integration period in seconds, uses float if < 1, integer if >= 1

DELETE – close running processes or list them
Usage:
DELETE TASK = <string> - Delete Post Processing Task
TASK = taskID - delete specified task
TASK = ALL - delete all tasks
TASK = NULL - list tasks

FFT – generate a spectrum plot of a data sequence

Usage: FFT [-hlt] vsr_hostname data_filename num_pts num_avg

arguments:

-h	Use a Hanning window on the fft input data points
-l	Process VSR data files stored in local unix format
-t {start_time}	ASCII string in YY/DDD/HH:MM:SS format that specifies the time to start processing the VSR data file (default: last record)
host	Name of VSR to get file from
data_filename	Name of BLS data file on the VSR to process
num_pts	Number of data points to use in the FFT
num_avg	Number of FFTs to average together

FILE – download a VSR data file and store it on the RAVI

FILE VSR_ID = <string> VSR_FILE = <string> LOCAL_FILE = <string>

- Copy VSR data at VSR_ID:_FILE to LOCAL_FILE

STOKES – generate stokes parameters from two data sequences

Usage: STOKES [-ahlt] vsr_host data_file1 data_file2 num_bin int_time

arguments:

-a	Align the sequences for maximum correlation (peak lag is centered)
-h	Use a Hanning window on the fft input data points
-l	Process VSR data files stored in local unix format
-t {time}	ASCII string in YY/DDD/HH:MM:SS format that specifies the time to start processing the data files (default: last record)
vsr_host	Name of VSR to get file(s) from
data_file1	Name of first BLS data file on the VSR to process
data_file2	Name of second BLS data file on the VSR to process
num_bin	Number of bins in the cross correlation spectrum
int_time	Integration period in seconds, uses float if < 1, integer if >= 1

QUIT- terminate execution

Usage:

QUIT

The Data Product

The data product of the RAVI is ASCII text files stored in **/ravi/data**, which is actually a symbolic link for **/data/ravi_data**. The commands that generate these files are FFT, CORR and STOKES. The FFT command generates filenames of the form:

FFT_NP<num_pts>_NA<num_avg>_<vsr_filename>.

The CORR command generates filenames of the form:

ACORR_NB<num_bins>_NI<integ_time>_<vsr_filename>
XCORR_NB<num_bins>_NI<integ_time>_<vsr_filename1>_<vsr_filename2>

for auto and cross correlations respectively.

The STOKES command generates filenames of the form:

STOKES_NB<num_bins>_NI<integ_time>_<vsr_filename1>_<vsr_filename2>.

The first data column in these files contains a floating point number representing the frequency in Hz. The second column of the FFT file represents the log amplitude spectrum. The second and third columns of the CORR files represent the log amplitude and phase spectrum respectively. The second through the fifth columns of the STOKES files represent I, Q, U and V respectively. Note that Stokes parameters in these files assume linear polarization.